## REMARKS

Claims 1-37 are pending in the above-identified patent application. Claims 1, 10, 18, 25, 30, 34 and 36 are independent.

The examiner objected to the drawings.

Applicant has amended FIG. 1 to properly label elements described in the specification at paragraph [0028]. No new matter was added.

The examiner rejected claims 18-24, 25-29 and 30-37 as having been directed to non-statutory subject matter. More particularly, the examiner argues that these rejected claims "fail to support a physical transformation or a practical application established by a useful, concrete, tangible result."

Applicant knows of no statutory requirement for claims to "support a physical transformation or a practical application established by a useful, concrete, tangible result." However, applicant is aware that the courts, i.e., the CAFC, has struck down any such a requirement. Applicant is also aware that the PTO has been informed of the CAFC rule.

Applicant's claims are clearly directed toward statutory subject matter. For example, claim 18, as amended, recites "a computer-implemented method of developing a software application comprising receiving a set of available patterns..., receiving requirements identified for a business process and mapping the identified requirements to patterns from the available patterns..." Claim 25, as amended, recites "an article comprising a machine-readable medium storing a data structure including definitions of a plurality of patterns for use in constructing a software application, the definitions of the plurality of patterns in the data structure comprising..." Claim 30, as amended, recites, " a computer-implemented method of developing a software application..." The courts have consistently considered these types of claims as being statutory subject matter under 35 U.S.C. §101. Further, the court held in In re Lowry that data structures are statutory subject matter under 35 U.S.C. §101. Accordingly, claims 18-24, 25-29 and 30-37 are proper under 35 U.S.C. §101.

The examiner rejects claim 26, under 35 USC §112, as being indefinite, asserting that it is unclear which "pattern" Claim 26, Line 1 is referencing. In response to this rejection, the applicant has amended Claim 26, Line 1 to address the examiner's concerns.

The examiner rejects claims 30-33, under 35 USC §112, as being indefinite, asserting that there is insufficient antecedent basis for "at least one work scenario user interface pattern". Applicant respectfully asserts that this rejection is improper. Specifically, the "at least one work scenario user interface pattern" referenced in the last line of claim 30 does not require antecedent basis, as the phrase "at least one work scenario user interface pattern" is being introduced for the first time in the last line of Claim 30. As claims 31-33 each directly depend from Claim 30, the applicant respectfully asserts that Claims 31-33 are also proper.

The examiner uses Barrett and Underwood to reject claims 1-37 as having been obvious.

Claims 1 and 10, as amended, recite "identifying pattern types corresponding to requirements for the software application to be developed, wherein the pattern types are identified from one or more of process patterns, user interface (UI) patterns, and generic object patterns," or similar language. Barrett and Underwood fail to teach or suggest this quoted claim feature.

Barrett teaches using a meta model to generate executable code:

> **[0009] According to the invention, there is provided a process for development of a software system, the process comprising the steps of defining a meta model and using the metal model to generate executable code... [Barrett, paragraph 0009]**

Barrett teaches away from applicant's claimed invention because Barrett specifically teaches using components in a specification phase:

> **[0013] In one embodiment, the meta model is generated from a design model arising from a specification phase, the design model specifying the system in terms of components and their interactions. [Barrett paragraph 0013]**

Underwood is no help. Underwood teaches a method of generating software based on business components [Underwood, Abstract] Nowhere does Underwood teach or suggest identifying pattern types corresponding to requirements for the software application to be developed, wherein the pattern types are identified from one or more of process patterns, user interface (UI) patterns, and generic object patterns, as claimed in claim 1. Underwood does not identify pattern types corresponding to requirements.

Underwood does teach class libraries:

> **Class libraries are very flexible. As programs grow more complex, more programmers are forced to reinvent basic solutions to basic problems over and over again. A relatively new extension of the class library concept is to have a framework of class libraries. This framework is more complex and consists of significant collections of collaborating classes that capture both the small scale patterns and major mechanisms that implement the common requirements and design in a specific application domain. They were first developed to free application programmers from the chores involved in displaying menus, windows, dialog boxes, and other**

**standard user interface elements for personal computers. [Underwood, col. X, lines y-z]**

Underwood teaches about a ReTA Activity framework design:

**This portion of the present description details the ReTA Activity framework design from the perspective of the application developer. The primary role of this framework is to provide services that support the "model view controller" (MVC) design pattern. In this pattern, the application implements a "separation of concern" among the user interface (view), logical unit of work (controller) and business components (model). Separating the user interface from the business logic increases reuse of the interface and the business component. In this design pattern, different types of interfaces can reuse the same model and the same interface can view different models. Another goal of separating presentation and storage responsibilities is to reduce the impact of change. For example, changing the user interface design should only impact the user interface components and not the business logic. Through modeling the "separation of concern" pattern, the ReTA Activity framework increases application maintainability and flexibility. It also encourages "best practice" coding standards. [Underwood, col. X, lines y-z]**

Underwood teaches auditing and logging:

**Application auditing and logging is often overlooked because it is less than glamorous, but it does provide security administrators with a crucial tool for monitoring use of an application. Good logs should be searchable for known or suspected patterns of abuse, and should be protected from alteration. Logs can monitor a virtual myriad of data, including access times, user IDs, locations from where the application was accessed, actions the user performed, and whether or not those actions were successfully completed. [Underwood, col. X, lines y-z]**

Underwood teaches monitoring for patterns of abuse:

**Before an incident can be responded to, it must first be detected. In the Net Centric environment, your firewall, routers, web servers, database servers, applications, and network management tools must be monitored to ensure they are working correctly and no violations have occurred. Monitoring packages can be configured to take different actions on a series of specified events, such as sending an email message if a log fills up; flashing an icon on a system administrator's screen if someone's user ID is disabled, or paging a network administrator if a link to the ISP goes down. Once this initial notification takes place, there should be escalation procedures to decide whom to notify next. For example, if the link to the ISP goes down, how long does one wait before notifying one's manager? One's users? In addition, not all monitoring needs to be reactive. There are proactive monitoring tools available which can detect patterns of abuse or failure which may lead to larger problems, and can help one detect those problems before they affect your users. [Underwood, col. X, line y-z]**

Still, nowhere does Underwood teach, suggest or even mention identifying pattern types corresponding to requirements for the software application to be developed, wherein the pattern types are identified from one or more of process patterns, user interface (UI) patterns, and generic object patterns. Accordingly, claims 1 and 10 is not obvious in view of Barret and Underwood, whether taken separately or in combination.

Claims 18, 25, 30, 34 and 36, as amended, recite "mapping the identified requirements to patterns from the available patterns by mapping process-related requirements to a corresponding business process pattern, wherein process-related requirements for which a business pattern is unavailable are mapped to a corresponding work process pattern, and process-related requirements for which a work process pattern is unavailable are mapped to a corresponding business action pattern," or similar language. Barrett and Underwood fail to teach or suggest this quoted claim feature.

As discussed about with respect to claims 1 and 10, Barrett and Underwood fail to teach or suggest any relationship between requirements and patterns. On the contrary, Barrett teaches using components in a specification phase. More specifically, Barrett merely teaches:

> **[0085] The component type is also used in the modeling of "Complex Component Type," which is a model of how a set of instances of component types can be bound together to behave as a single component (FIG. 6). Such a complex component will behave as if it were a single component. The Complex component type modeling describes the required and provided interfaces of the type, the types of sub-components that will go to make up the aggregate, and a modeling of how the interface requirements and provisions of these sub-components map to the requirements and provisions of the composite whole. The purpose of this is to support the modeling of a typed subsystem as an assembly of other subsystems. [Barrett, paragraph 0085]**

Underwood merely teaches patterns of abuse. Since Barrett and Underwood fail to teach of suggest any relationship between requirements and patterns, no combination of Barrett and Underwood can teach or suggest any relationship between requirements and patterns. Accordingly, claims 18, 25, 30, 34 and 36 are not rendered obvious by Barrett and Underwood.

It is believed that all of the pending claims have been addressed. However, the absence of a reply to a specific rejection, issue or comment does not signify agreement with or concession of that rejection, issue or comment. In addition, because the arguments made above may not be exhaustive, there may be reasons for patentability of any or all pending claims (or other claims) that have not been expressed. Finally, nothing in this paper should be construed as an intent to concede any issue with regard to any claim, except as specifically stated in this paper, and the amendment of any claim does not necessarily signify concession of unpatentability of the claim prior to its amendment.

Respectfully submitted,

[illegible]

BRIAN J COLANDREO
REG NO 42,427 FOR
Kenneth F. Kozik
Reg. No. 36,572

Date: 26 September 2006

Holland & Knight LLP
10 St. James Avenue
Boston, MA 02116
Tel. 617.573.5879
Fax 617.523.6850

# 4049613_v1